Testing Report

# Purpose (of this report)

The purpose of this paper is to report the testing results applied to the Calendar Project from Spring 2016 semester.

# Scope

## Software

Software I used to test the software included Visual Studio, Microsoft Word, and HeidiSQL.

## Hardware

To test this project, I used a Toshiba Satellite laptop and a Lenovo Thinkpad laptop.

## People

I (Ethan Vaughan) performed the testing on this software.

# References/Documents

Refer to the “Testing Log” document for all the test case results and how I performed the tests.

# Testing Plan

## Testing Strategy and Methods

I began testing with the Use Case based testing. I tested each use case with different combinations of input. When I found an error, would locate the function that is first called after performing the GUI operations. I would then use a WBT technique to figure out what caused the error. I finished with testing the Event class (non-GUI class).

The testing techniques used were Use Case Based testing, BBT Boundary Value Analysis, BBT Equivalence Partitioning, WBT Basis Path Testing, and WBT Condition Testing.

## Testing Schedule

March 27 – Project introduction (in class)

4:40 PM – 5:55 PM: Project was introduced to whole class & posted on Blackboard

March 28 – Schedule preparations

8:00 PM – 8:30 PM: Setup schedule

March 29 – Planning day

6:00 PM – 6:30 PM: Develop plan to test use case

March 30 – Begin testing

9:00 AM – 9:30 AM: Finalize schedule

11:00 AM – 12:15 PM: Begin use case-based testing for Use Case 1

April 1 – Begin Testing Use Case 1

11:00 AM – 10:45 PM: Test Use Case 1

April 6 – Finish testing Use Case 1; begin testing Use Case 2 and Use Case 3

12:15 PM – 1:00 PM: Finish testing Use Case 1

8:00 AM – 6:45 PM: Test Use Case 2

7:00 PM – 9:00 PM: Test Use Case 3

9:00 PM – 9:30 PM: Test Use Case 4

9:30 PM – 10:15 PM: Test Use Case 5

April 10 – Test Event class

11:30 AM – 12:10 PM: Test getEventList function

1:00 PM – 2:00 PM: Test getMonthlyEventList function

2:15 PM – 2:45 PM: Test checkStartEndTimeConflict function

*A note about the schedule: You may be wondering why the work I did on this project was so uneven (I did most of the work during the first two weeks of April, yet the project wasn’t due until April 30). This is because I knew I would have lots to do in my other classes towards the end of the month. I wanted to get the project done as soon as possible so I could have more time to study for finals and focus on other major assignments as they came up.*

# Testing Log

## Indexing Number/Date/Time/Test Items/Testing Method/Test Case(s)/Test Data/Result/Deficiencies/Location(s)/Indexing number of further test(s)

### Use Case Based Testing

The use cases for the project, as described in “Individual Project Spring 2016”, are as follows:

* Add an event to the calendar, including conflict check of two events. (UC1)
* Delete an event from the calendar. (UC2)
* Edit an event on the calendar. (UC3)
* View an event. (UC4)
* View a monthly-based event list. (UC5)

I have numbered them UC1-UC5 for reference in the Use Case Based testing.

**Use Case 1**

ID: UC1

Date: 04/01/2019 - 04/06/2019

Testing Method(s): Use Case Based testing; WBT Condition Testing

Name: Add an event to the calendar, including conflict check of two events.

Actor: User who wants to add an event to the calendar.

Preconditions: User was able to start the system. Access the remote database; XAMPP is running.

|  |  |
| --- | --- |
| Actor: User | System: Calendar Program |
| 1. TUCBW the user clicking on the “New” button to add a new event. | 2. System allows user to input information about the event. |
| 3. The user enters the event related info. | 4. System verifies info. |
| 5. User clicks “Save” button. | 6. System adds the event to the database. |

The above shows the Primary Scenario (the normal handling of Use Case 1). However, there could be other special cases (Secondary Scenarios). These are displayed below by use of a table showing all user input that is required and the normal case input. Beside the normal cases will be alternative cases (basically, input that would cause a Secondary Scenario to occur). In the far right column is the exception that should get thrown for the corresponding alternative case.

|  |  |  |  |
| --- | --- | --- | --- |
| User Input | Normal Case | Alternative Case | Exceptions |
| Event title | Valid | No title given | Allow empty event title. |
| Date | Valid | NA | Invalid |
| Start time | Before end time | After end time | Message box; do not let user add event. |
| End time | After start time | Before start time | Message box; do not let user add event. |
| Description | Valid | NA | Invalid |

After I created the chart for identifying Secondary Scenarios, I designed test cases to test each scenario.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index Number | Scenario | Event title | Date | Start time | End time | Description | “Save” button click | “Ignore” button click | Expected results |
| TC1 | Primary scenario | “test” | 2019-04-02 | 10:00 | 11:00 | “test” | Clicked | Not clicked | Display confirmation |
| TC2 | No title given | NULL | 2019-03-30 | 10:00 | 11:00 | “test” | Clicked | Not clicked | Allow empty title. |
| TC3 | Start time comes after end time | “test” | 2019-03-30 | 12:00 | 11:00 | “test” | Clicked | Not clicked | Display error message |
| TC4 | “Ignore” button clicked | “test” | 2019-03-30 | 10:00 | 11:00 | “test” | Not clicked | Clicked | Remove the entered information from view; take user back to start |
| TC13 | Start time equals end time | “test” | 2019-04-07 | 1:00 | 1:00 | “test” | Clicked | Not clicked | Event should be passed to the database (allowed since events can happen instantaneously) |
| TC14 | Event date for event in the past | “test” | 2019-02-01 | 3:00 | 4:00 | “test” | Clicked | Not clicked | Should be allowed |
| TC15 | Event date for event today | “test” | 2019-04-08 | 3:00 | 4:00 | “test” | Clicked | Not clicked | Should be allowed |
| TC16 | Event date for event in the future | “test” | 2019-04-05 | 1:00 | 1:30 | “test” | Clicked | Not clicked | Should be allowed |

PROBLEMS:

* TCA date could not be selected after user had already clicked the “New” button.
* Event information was not blanked out when use clicked “New”, entered event-related information, and clicked “Ignore”.

Based on these results, further testing was necessary to identify the root cause of the errors and what steps the programmer(s) must take to correct the problems.

First, I worked on finding solutions to the previous date problem (user not being able to select a date in the past when creating a new event). Below screenshot shows the issue. As you can see, I clicked the “New” button in an attempt to select a new event. The fields popped up requesting my input. I clicked “April 3, 2019” as the date, and the Date text box blanked itself (instead of populating with 2019-04-03).

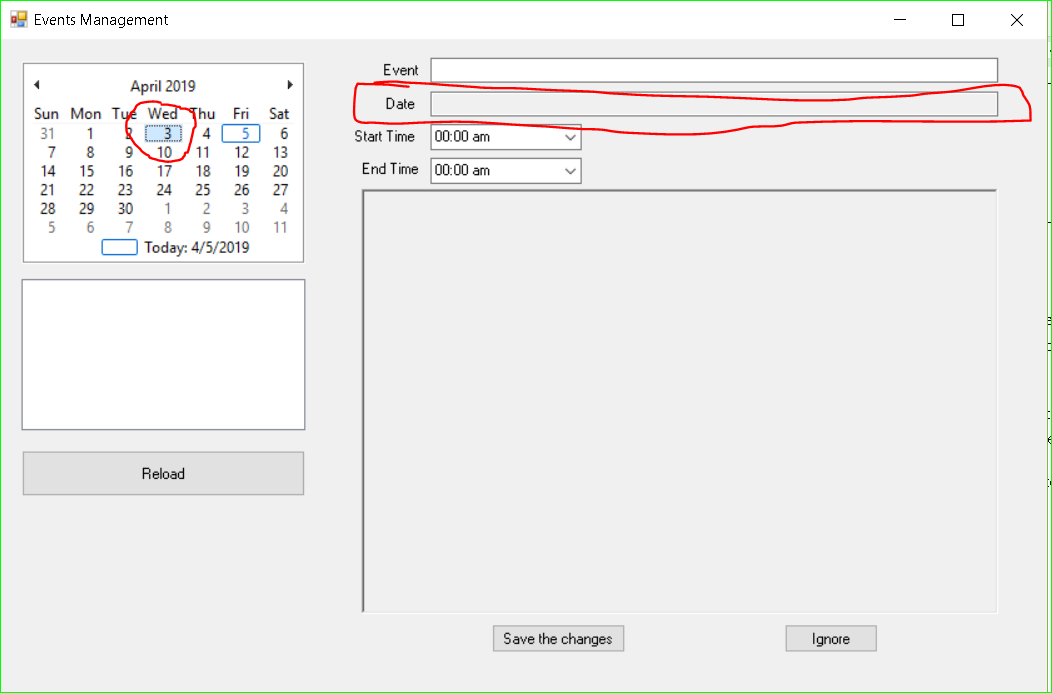


Figure 1 TC14

So, the first thing I needed to do was find out what function gets called when I select a new date in the calendar. (Fortunately, Visual Studio is quite handy for this.)

I discovered that a function called void monthCalendar1\_DateChanged(object sender, DateRangeEventArgs e) is the first unit to get called when a calendar’s date is changed. (Note: the input parameters, sender and e, are automatically put in by Visual Studio when the function is created. They can be disregarded during testing.)

This function contains a foreach loop and an if statement, thus it made a good candidate for WBT Condition Testing. I found that I needed to pass two types of inputs to the function in order to test all branches: an empty event list and a nonempty event list. How does a user pass each to the system? By selecting days that contain events and days that don’t contain events. Below is the flow chart for the function.



FURTHER TEST CASES:

* TC17 (user picks a day without any events while trying to add an event to the calendar).
* TC18 (user picks a day with events while trying to add an event to the calendar)

RESULTS OF FURTHER TESTING:

TC17 revealed that an event’s information (title, description, date, start time, end time, description) is automatically loaded into the input fields when a date is selected that contains events. Upon inspecting the code, it appears that the first event in the list of events returned from getEventList is the one that is loaded into the input fields. (The screenshot shows that the event-related information for the first event that day was automatically loaded into the input fields when a day was selected after I clicked “New”.)

RECOMMENDATIONS TO PROGRAMMERS: If the user is trying to create a new event, do not load other event information into the input fields when a different date is selected. Place logic to check for the new event mode inside the monthCalendar1\_DateChanged method.

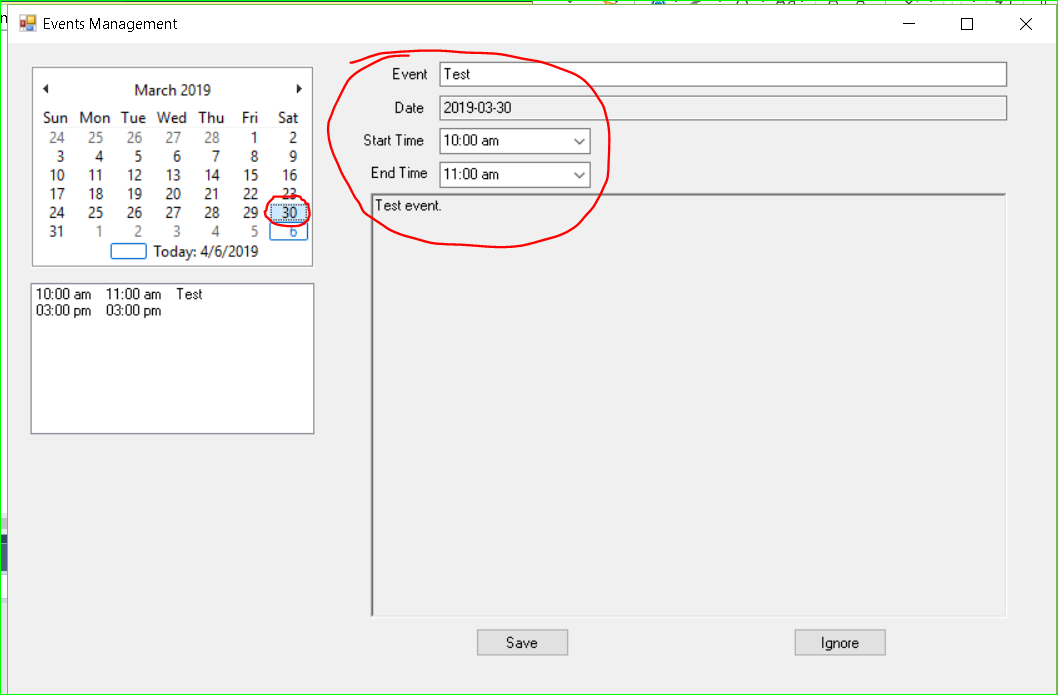


Figure 2 TC17

TC18 revealed more serious problems with the application than TC17. When I tried selecting an event for a previous date that contained no events, no date was placed into the Date text box. As this text box is not editable directly by the user (i.e. I can’t type in a date), there was nothing more I could. I simply could not add a date.

The problem is again causes within the monthCalendar1\_DateChanged function. This function only inserts a date into the Date text field if it can find an event on the specified day.

RECOMMENDATIONS TO PROGRAMMERS:

Allow user to select a date for new events. Do not let the program rely on the date already containing events.

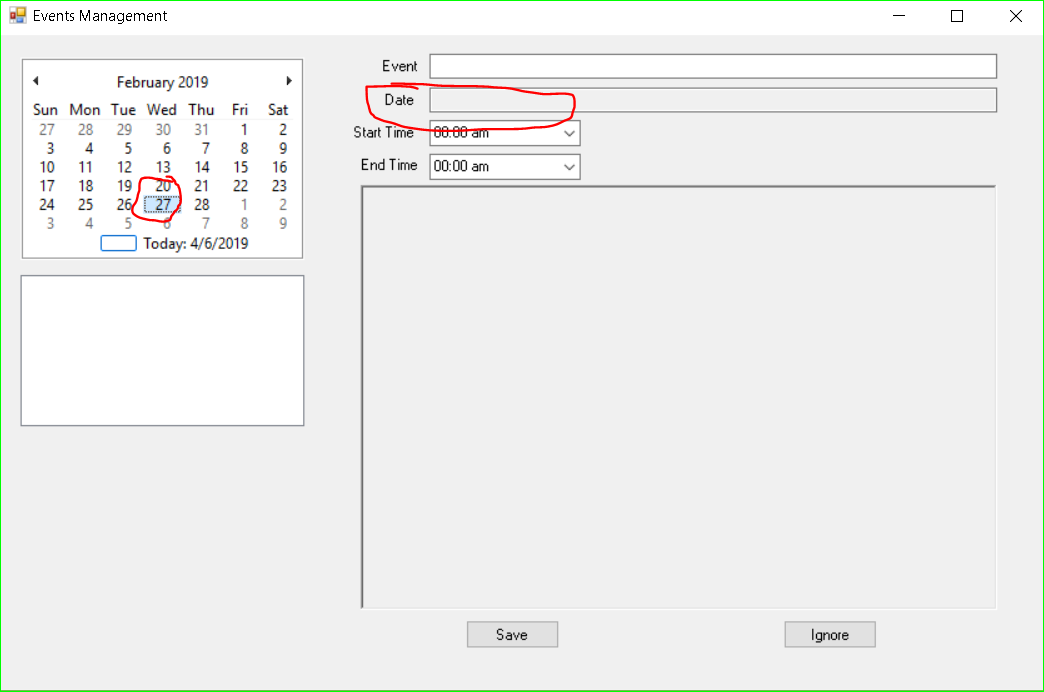


Figure 3 TC18

**Use Case 2**

ID: UC2

Date: 04/06/2019

Testing Method(s): Use Case Based testing; WBT Basis Path Testing

Name: Delete an event from the calendar.

Actor: User who wants to delete the event.

Preconditions: User was able to start the system. Access the remote database; XAMPP is running.

|  |  |
| --- | --- |
| Actor: User | System: Calendar Program |
| 1. TUCBW the user selecting an event to delete. |  |
| 2. The user clicks the “Delete” button. | 3. System puts the events details in the entry boxes and asks user to confirm delete. |
| 4. User clicks “Confirm to delete”. | 5. System removes the event from the screen. |

Below are possible Secondary Scenarios.

|  |  |  |  |
| --- | --- | --- | --- |
| User Input | Normal Case | Alternative Cases | Exceptions |
| “Done” button click | Valid | No event selected. | Delete option not available |
| “Ignore” button click | Valid | NA | Invalid (not possible to click the button if an event was not selected) |
| Date selected | Valid | No date | This is impossible; GUI will not permit it |
| Event title | Valid | Empty | Invalid |
| Event start time | Valid | After end time | Impossible (events cannot be created with start time after end time) |
| Event end time | Valid | Before start time | Impossible (events cannot be created with start time after end time) |
| Event description | Valid | Empty | Invalid |

After I created the chart for identifying Secondary Scenarios, I designed test cases to test each scenario.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index Number | Scenario | “Delete” button click | “Ignore” button clicked | Date selected | Event title | Event start time | Event end time | Event description | Expected results |
| TC5 | Primary scenario | Clicked | NA | 2019-03-30 | “tc5” | 10:00 | 11:00 | “tc5” | Display confirmation |
| TC6 | No event selected | Clicked | NA | NA | NA | NA | NA | NA | Delete button not clickable. |
| TC19 | No title | Clicked | NA | 2019-03-30 | NULL | 10:00 | 11:00 | “tc19” | Event should be removed from the database. |
| TC20 | Event description empty | Clicked | NA | 2019-03-30 | “tc20” | 10:00 | 11:00 | NULL | Event should be removed from the database. |
| TC21 | “Ignore” button clicked | NA | Clicked | 2019-03-30 | “tc21” | 10:00 | 11:00 | “tc21” | GUI should take user out of delete mode. |

PROBLEMS:

* TC20 revealed that events without titles apparently could not be deleted. Further testing using WBT Basis Path testing was performed to find the specific location of the error(s).

I first had to find out which function is called when the “Confirm to delete” button is clicked. I discovered that button8\_Click is the function that is called when a user wishes to delete an event. So, I constructed a flow chart of that function (see Figure 4).



Figure 4 button8\_Click Flow Chart

As you can see, there are nine predicate nodes within this function. I therefore designed ten possible basis paths and data to test them.

|  |  |  |
| --- | --- | --- |
| Test case (index number) | Basis Path | Explanation |
| TC22 | 1-2-3-23-4-5-8-7-9-20-22 | Event is deleted; nothing else appears on the event list. |
| TC23 | 1-2-3-23-4-5-6-7-9-20-22 | Event is deleted; no more events to display for selected day. |
| TC24 | 1-2-3-23-4-5-6-7-9-12-13-14-16-9-10-21-22 | Event is deleted; display the next event. |
| TC25 | 1-2-3-23-4-5-6-7-9-12-13-14-16-17-9-10-20-22 | Event is deleted; display the next event. Next event has same title as deleted event. |
| TC26 | 1-2-3-23-4-5-6-7-9-12-13-14-16-17-18-9-10-20-22 | Event is deleted; display the next event. Next event in sequence has same title and start time as deleted event. |
| TC27 | 1-2-3-23-4-5-6-7-9-12-13-14-16-17-18-19-9-10-20-22 | Event is deleted; display the next event. Next event in sequence has same title, start time, and end time as deleted event. |
| TC28 | 1-2-3-23-4-5-6-7-9-12-13-14-16-9-10-20-22 | Event is deleted; display next event for monthly events. |
| TC29 | 1-2-3-23-4-5-8-7-9-12-15-14-16-9-10-20-22 | Event is deleted; display the next event for daily events. |
| TC32 | 1-2-3-23-22 | Event title is null. |

RESULTS OF FURTHER TESTING:

TC32 revealed that an event cannot be deleted if it has a title that is null.

RECOMMENDATIONS TO PROGRAMMERS:

The textBox1.Text != "" clause from the first if statement inside the button8\_Click function is the problematic piece of code. The programmers can either allow users to delete events that don’t have a title, or don’t allow users to enter events without titles.

**Use Case 3**

ID: UC3

Date: 04/06/2019

Testing Method(s): Use Case Based testing

Name: Edit an event on the calendar.

Actor: User who wants to edit an event on the calendar.

Preconditions: User was able to start the system. Access the remote database; XAMPP is running.

|  |  |
| --- | --- |
| Actor: User | System: Calendar Program |
| 1. TUCBW the user clicking on the “Modify” button to edit an event. | 2. System allows user to change information about the event. |
| 3. The user enters the updated event related info. | 4. System verifies info. |
| 5. User clicks “Save the Changes” button. | 6. System adds the changes to the database. |

Below are possible Secondary Scenarios to Use Case 3.

|  |  |  |  |
| --- | --- | --- | --- |
| User Input | Normal Case | Alternative Case | Exceptions |
| “Modify” button click | User clicks “Modify” button | No event selected. | Nothing happens (as no event to modify is selected/displayed) |
| “Ignore” button click | User clicks “Ignore” button | NA | Invalid |
| Event title | Valid | No title given | Invalid |
| Date | Valid | Different date selected while trying to modify event. | Event should remain in view. |
| Start time | Before end time | After end time | Message box; do not let user add event. |
| End time | After start time | Before start time | Message box; do not let user add event. |
| Event description | Valid | NA | Invalid |

After I created the chart for identifying Secondary Scenarios, I designed test cases to test each scenario.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index Number | Scenario | “Save the changes” button click | “Ignore” button click | Event title | Date | Start time | End time | Event description | Expected results |
| TC7 | Primary scenario | Clicked | Not clicked | “test” | 2019-03-30 | 3:00 | 3:00 | “test” | Display confirmation; show updated info |
| TC30 | “Ignore” button clicked | Not clicked | Clicked | “test” | 2019-03-30 | 3:00 | 4:00 | “test” | Cancel modifications, if any. |
| TC8 | No title given | Clicked | Not clicked | NULL | 2019-03-30 | 10:00 | 11:00 | “test” | Allow an empty title. |
| TC9 | Start time comes after end time | Clicked | Not clicked | “test” | 2019-03-30 | 12:00 | 11:00 | “test” | Display error message |
| TC31 | Date changed for event. | Clicked | Not clicked | “test” | 2019-03-30 | 12:00 | 11:00 | “test” | Change the event’s date to the newly selected date. |

PROBLEMS:

* TC31 revealed that the date could not be changed for the selected event.

This issue is the same as the one discovered in TC18. The date being selected automatically results in the text and combo boxes being filled with the first event for that day.

RECOMMENDATIONS TO PROGRAMMERS:

Allow the user to change an event’s date in edit mode.

**Use Case 4**

ID: UC4

Date: 04/06/2019

Testing Method: Use Case Based testing

Name: View an event.

Actor: User who wants to view an event.

Preconditions: User was able to start the system. Access the remote database; XAMPP is running.

|  |  |
| --- | --- |
| Actor: User | System: Calendar Program |
| 1. TUCBW the user clicking on an event. | 2. The system displays the event’s info. |

Below table will help identify any Secondary Scenarios.

|  |  |  |  |
| --- | --- | --- | --- |
| User Input | Normal Case | Alternative Case | Exceptions |
| Select an event from the list | Valid | NA | NA |
| Date | Select a date from the calendar | Date without any events | Display an empty list of events |

After I created the chart for identifying Secondary Scenarios, I designed test cases to test each scenario.

|  |  |  |  |
| --- | --- | --- | --- |
| Index Number | Scenario | Date | Expected results |
| TC10 | Primary scenario | Valid | Display the event |
| TC33 | Select a date without events | Valid | Display an empty list of events |

Use Case 4 did not provide much opportunity for Secondary Scenarios.

PROBLEMS:

None found.

**Use Case 5**

ID: UC5

Date: 04/06/2019

Testing Method(s): Use Case Based testing; BBT Boundary Value Analysis

Name: View a monthly-based event list.

Actor: User who wants to view the monthly-based events.

Preconditions: User was able to start the system. Access the remote database; XAMPP is running.

|  |  |
| --- | --- |
| Actor: User | System: Calendar Program |
| 1. TUCBW the user selecting a month to view the events for. |  |
| 3. The user click on the “Display Monthly Events” button. | 4. System displays a summary list of the month’s events. |

As this use case involves selecting a day of a certain month and then clicking the “Display Monthly Events” button, I decided to use BBT Boundary Value Analysis. That way, I can test that the program correctly shows monthly events for a month if I selected the last day or the first day.

The below table will help identify any Secondary Scenarios.

|  |  |  |  |
| --- | --- | --- | --- |
| User Input | Normal Case | Alternative Case | Exceptions |
| Calendar date | Valid | No events occurring in selected month | Empty list displayed |
| “Display Monthly Events” button click | Valid | NA | NA |

Below table shows the test cases designed to test the Secondary Scenarios and the Boundary Value Analysis tests.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Index Number | Scenario | Date | “Display Monthly Events” button click | Expected results |
| TC11 | Primary scenario | March 21, 2019 | Clicked | List of events |
| TC12 | No events for selected month | June 12, 2019 | Clicked | Empty list of events |
| TC35 | First day of the month selected. | January 1, 2019 | Clicked | List of events for January, 2019 |
| TC36 | Last day of the month selected | December 31, 2018 | Clicked | List of events for December, 2018 |

PROBLEMS:

No problems were encountered. All the test cases passed.

### Integration Testing

Since this is an Object-Oriented system, the type of integration testing will not involve top-down or bottom-up testing. However, I had to figure out what the relationships are among the methods and classes so I could perform testing in the correct order (test inheritance, then aggregation, and finally association). The GUI has already been tested for this application. So I needed to test the Event class next.

The first method I considered was the constructor (***Event(string s1, string s2, int s3, int s4, string s5)***), as all other methods within the Event class relied upon this method. The purpose of this method, as it is a constructor, is to initialize some variables that will be used throughout the class. There is no return value or much of anything that happens in this method; thus, I concluded that applying specific test cases was not necessary.

The next group of methods I considered were the getters (***getDate(), getTitle(), getStartTime(), getEndTime(), getContent(), getEventList(string dateString), getMonthlyEventList(string dateString)***). Many of these methods are relatively simple (they return a value and that is all the function does), such as getDate(), getTitle(), getStartTime(), getEndTime(), getContent(), and they contain only one statement (a return statement), so I did not perform any tests on these functions.

**getEventList(string dateString)**

The getEventList(string dateString) method is supposed to return a list of events for the specified date. I found that this method was called at such times as when a user first logs in or when a new date is selected.

It was best that I tested this method using BBT Equivalence testing, as there are technically only two domains of input: dates that contain events and dates that don’t contain events. (NOTE: this method accepts a string as input. Methods that call getEventList will not pass in invalid strings.) Thus, I tested the method by logging in, clicking on a date that contains an event, clicking on a date that does not contain an event. The method returned correct values in both cases. (Refer to index numbers 37 and 38 of the testing log.)

**getMonthlyEventList(string dateString)**

This method is called when the user clicks the “Display Monthly Events” button. It returns a list of events for the month. This method accepts a datestring of the form “YYYY-MM”. Knowing this, it is best to test the different paths the function could take before returning the list of events (WBT - Data Flow Testing). Months that contained no events (testing log index number 39) and months that contained events (index number 40) were selected.

Next, I needed to test the Form1 class. The purpose of this class is to control the GUI. When a button is clicked by the user or a new date is selected, functions within this class are immediately called.

The first function I tested was the checkStartEndTimesConflict(int startTimeIndex, int endTimeIndex) function. This function ensures that the start time comes before the end time for an event. Using BBT Boundary Value Analysis, I can check that the message box pops up if mismatched start/end times are entered.

Since the checkStartEndTimesConflict function should prevent user from entering a start time that is occurs after an end time, I will need to design **two test cases**: one with start time less than end time (the normal case – see index number 41) and another with start time after than end time (see index number 42). The way this function works is by taking advantage of the fact that times are organized in half-hour increments. Every increment is placed in a drop-down menu and corresponds to an index. Thus, if the startTimeIndex is less than the endTimeIndex, then the event works.

# Summary/Conclusion

This program is not far from being a well-designed app. All the primary scenarios seem to work. Users can add events to the calendar. It is user-friendly. However, the program is not production-ready. I found several significant problems when performing tests on the secondary scenarios. The insertion of a date into the date of the event seems to contain some flaws. Also, the user not including a title and/or event description seems to cause a problem, as these are both used to identify the event. If the programmer follows my advice in the areas outlined in this document, then the Calendar Program should be ready for production.